openGauss AI特性创新实践课



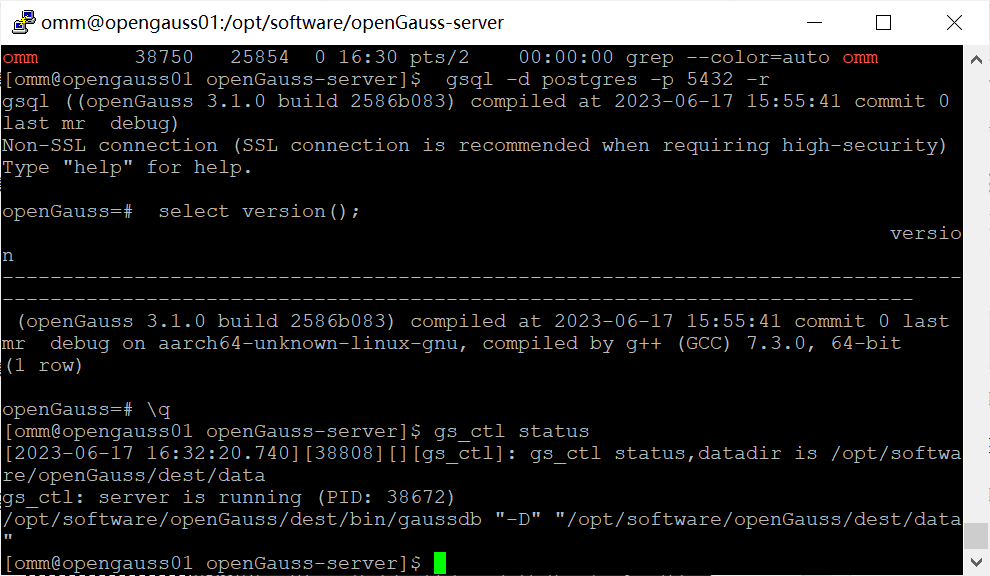
华为技术有限公司

# 关卡一、openGauss数据安装及基本操作

openGauss数据安装及基本操作, 作业提交任务如下：

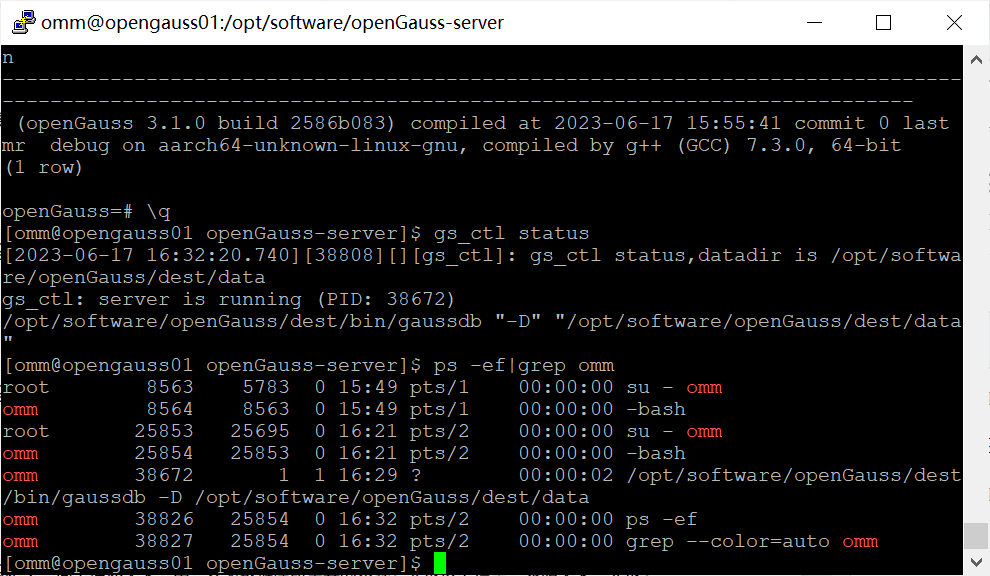
任务一：数据库状态验证

1. 查询数据库状态成功截图



任务二：数据库服务进程验证

1. 查看数据库服务进程截图（包含数据库服务器的主机名）



实验思考题：为什么需要通过源码编译，安装数据库？

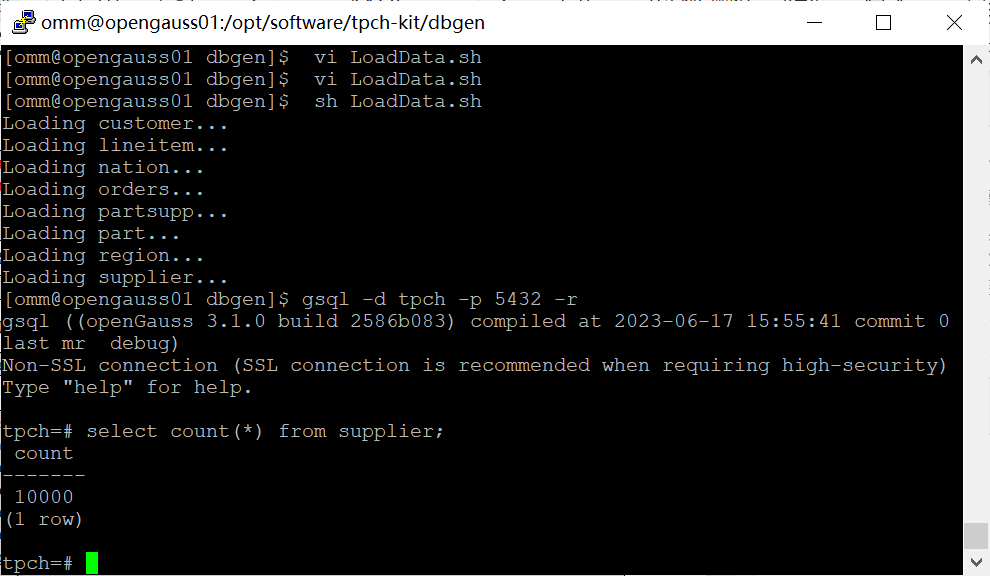
版本定制：源代码编译允许你根据特定需求和配置进行个性化定制。你可以选择启用或禁用特定的功能、插件或模块，以便数据库满足你的特定需求。这种灵活性在特定场景下非常重要。  
  
最新功能和改进：通过源代码编译，你可以获得最新的版本、功能和改进。在发行预编译的二进制包之前，可能会有一段时间的延迟。通过源代码编译，你可以立即获得最新的功能、安全补丁和性能优化。  
  
平台兼容性：预编译的二进制包通常是为特定平台和操作系统构建的。如果你的环境与支持的二进制包不匹配，或者你使用的是非常古老或非常新的平台，那么通过源代码编译可以确保数据库能够在你的环境中运行。  
  
自定义配置和优化：源代码编译允许你根据你的硬件和软件环境进行优化和调整。你可以选择适合你的硬件架构的编译选项，以提高性能和效率。此外，你还可以根据数据库的使用情况和负载模式进行特定的配置和调整。

# 关卡二、openGauss数据导入及基本操作

任务一：数据初始化验证

1. 查询supplier表的行数，并将结果进行图：

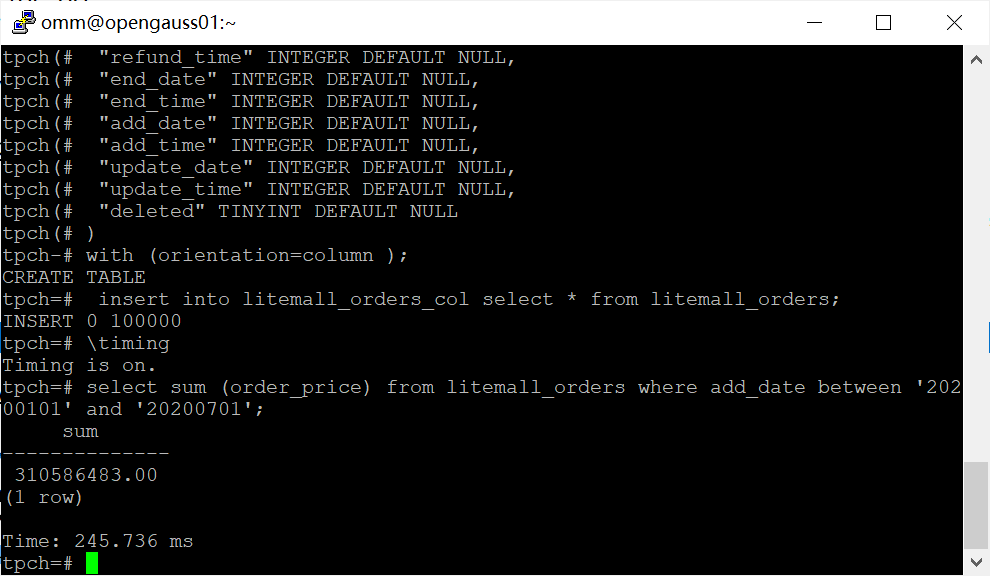
select count(\*) from supplier;;



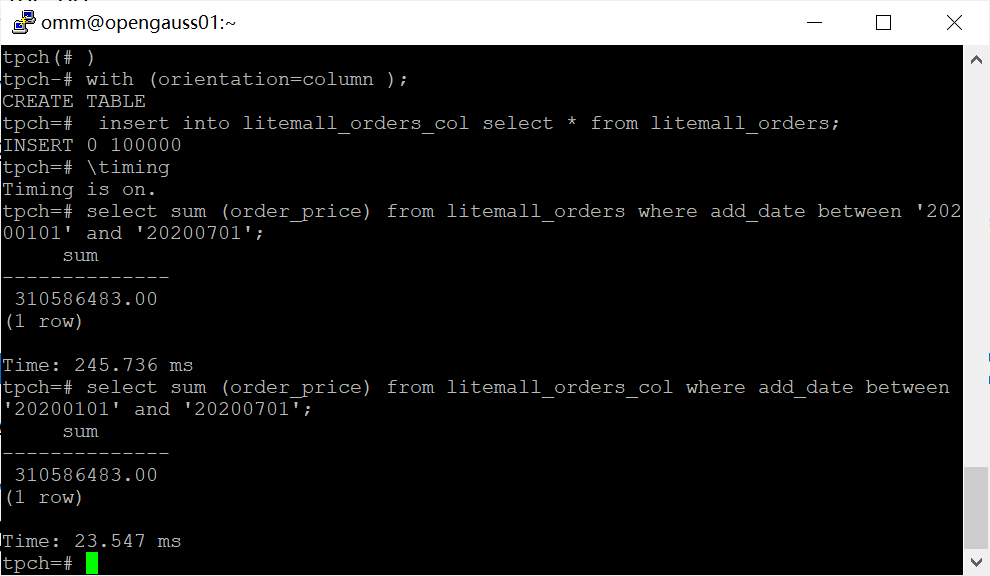
任务二：行存表与列存表执行效率对比

1. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的总和查询，并对比执行效率截图

select sum (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

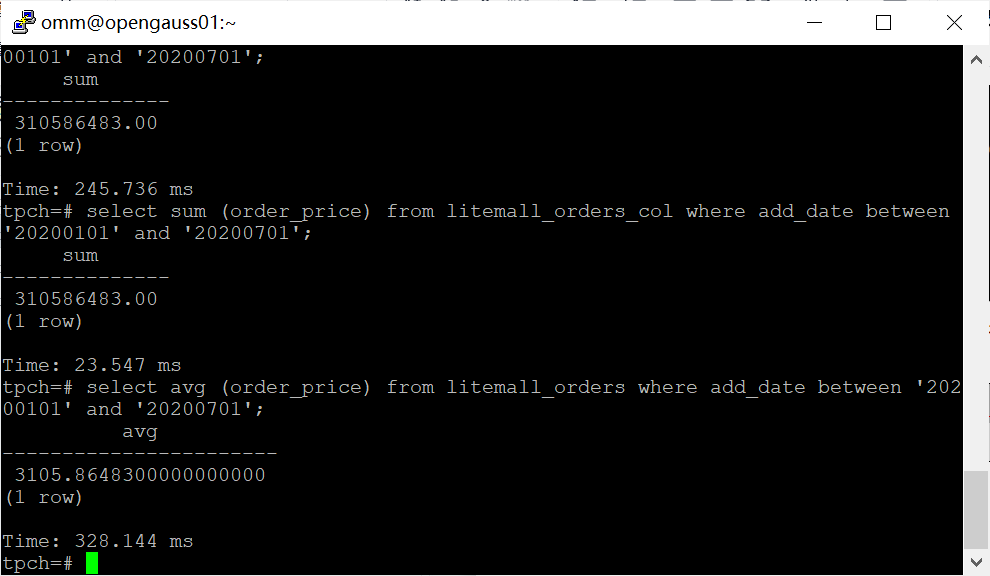


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

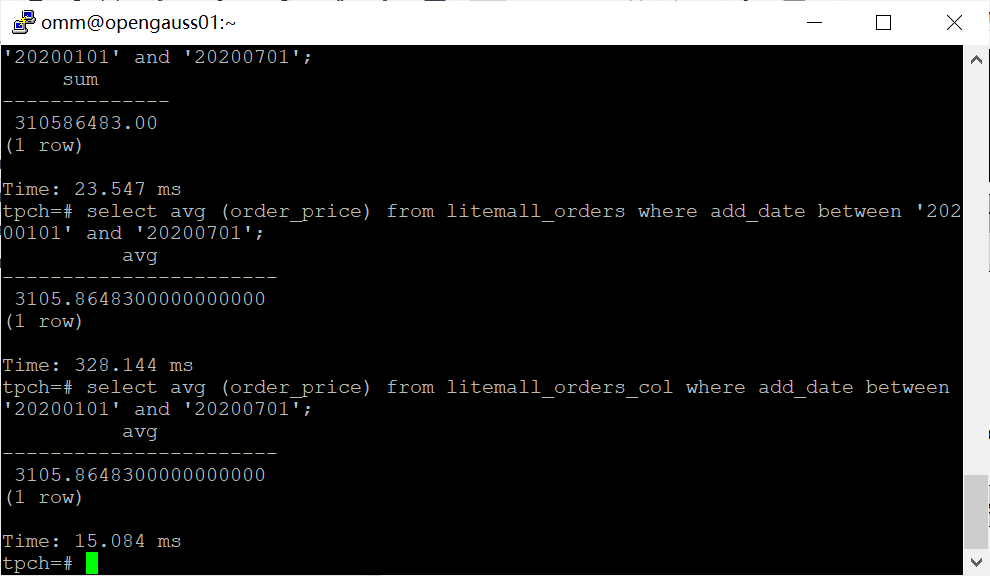


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

select avg (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

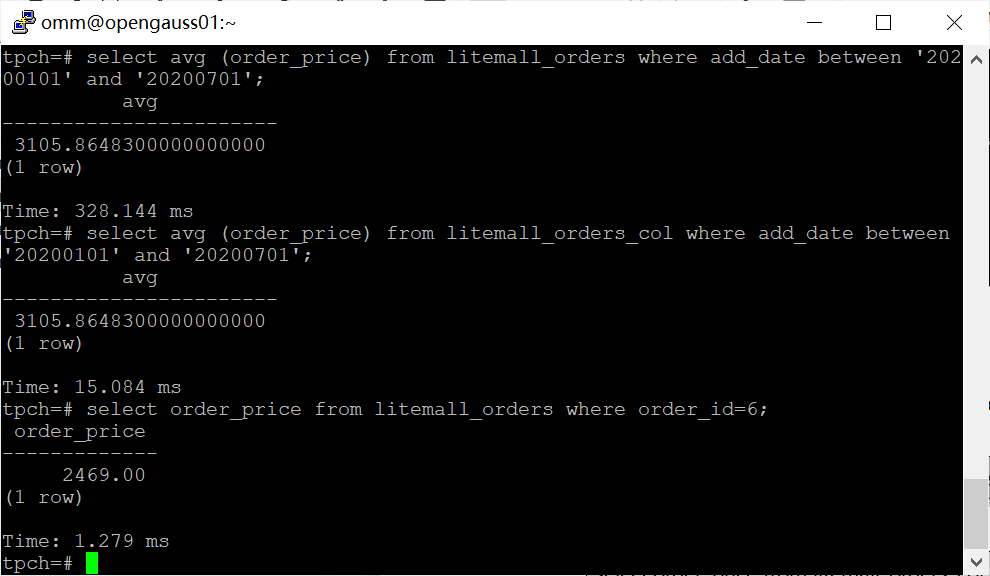


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

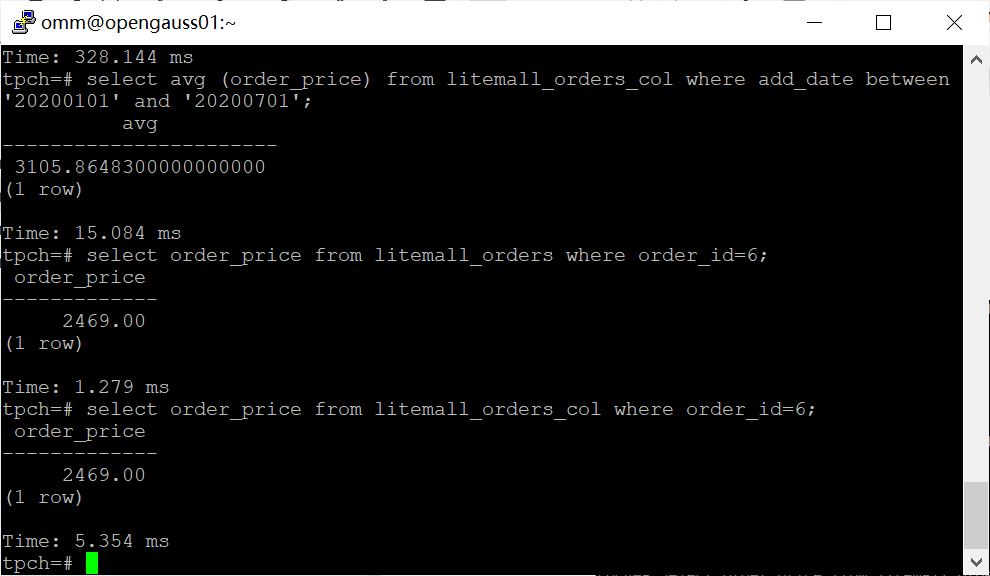


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

select order\_price from litemall\_orders where order\_id=6;

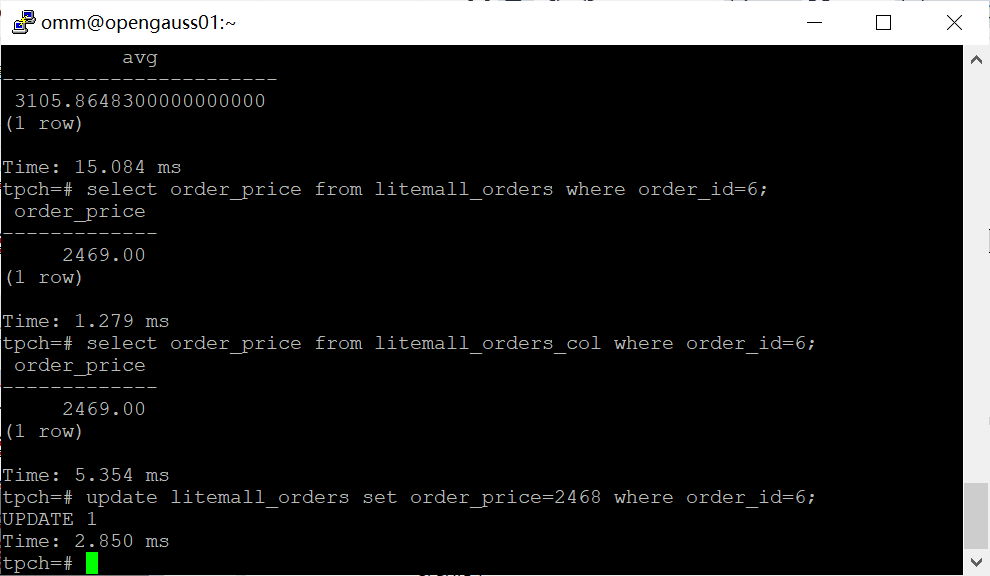


select order\_price from litemall\_orders\_col where order\_id=6;

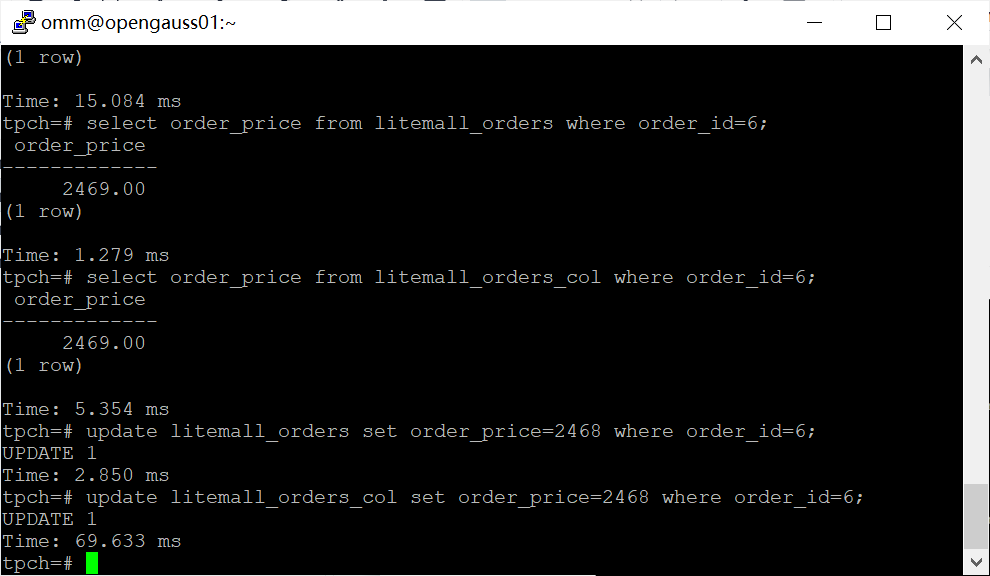


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

update litemall\_orders set order\_price=2468 where order\_id=6;



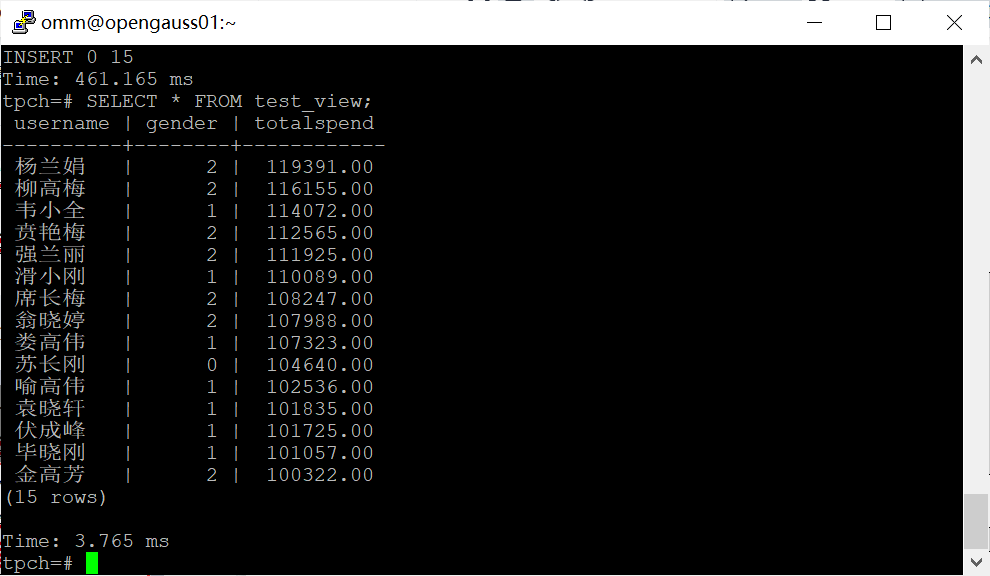
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

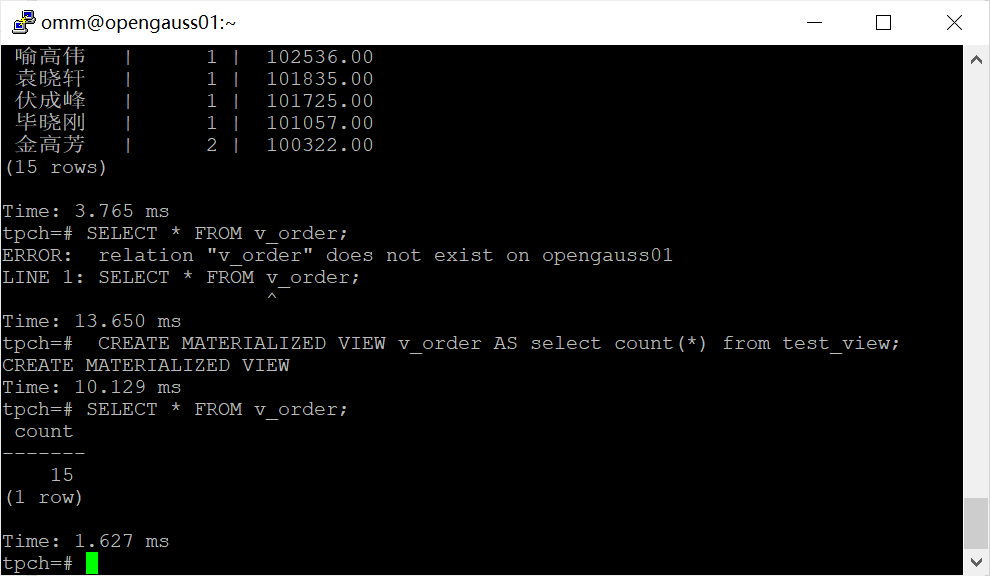
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



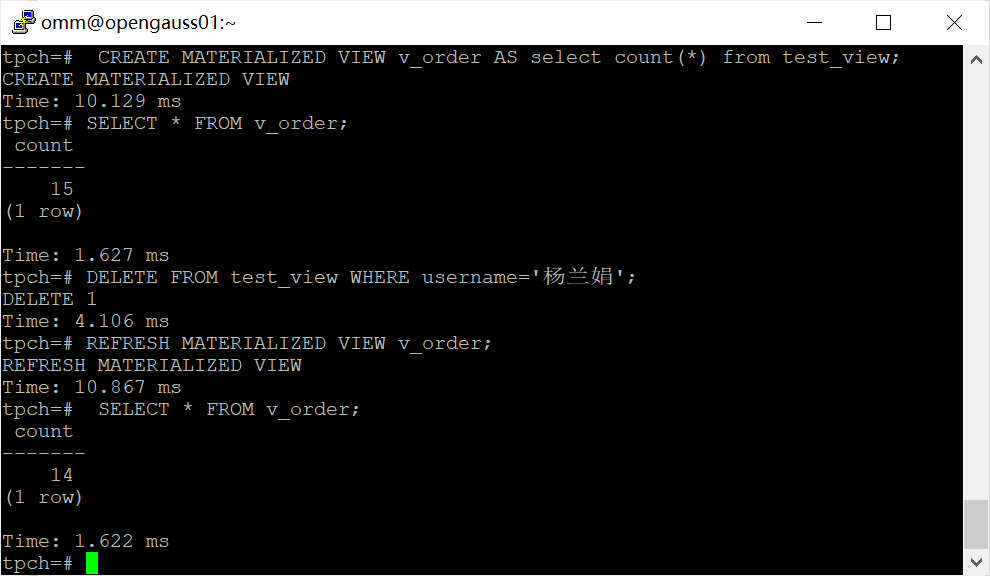
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



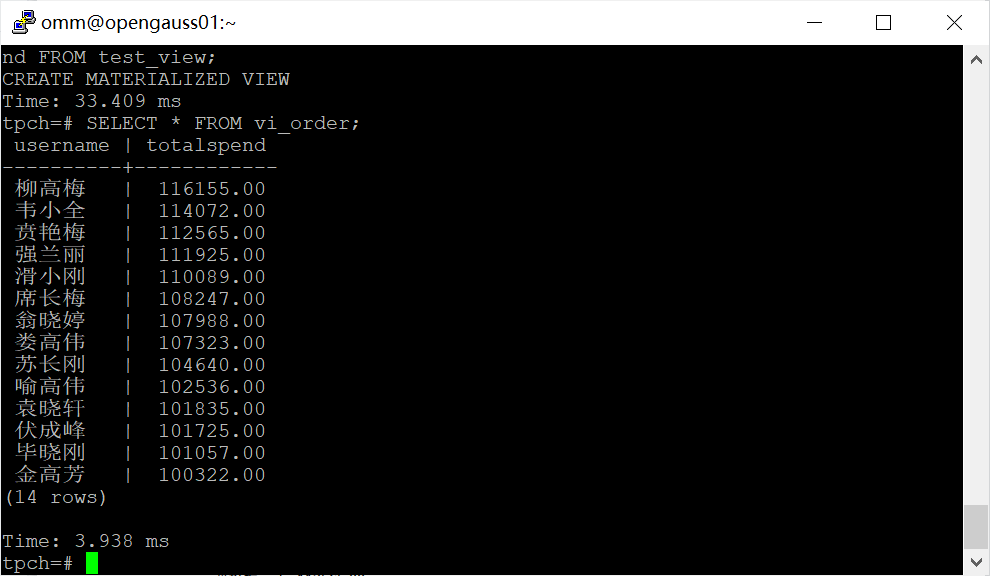
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



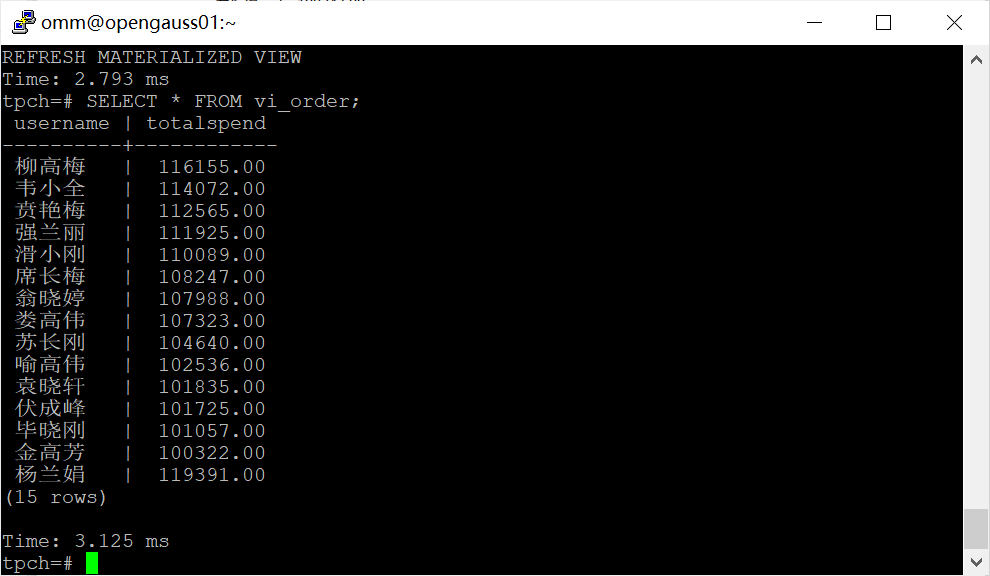
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

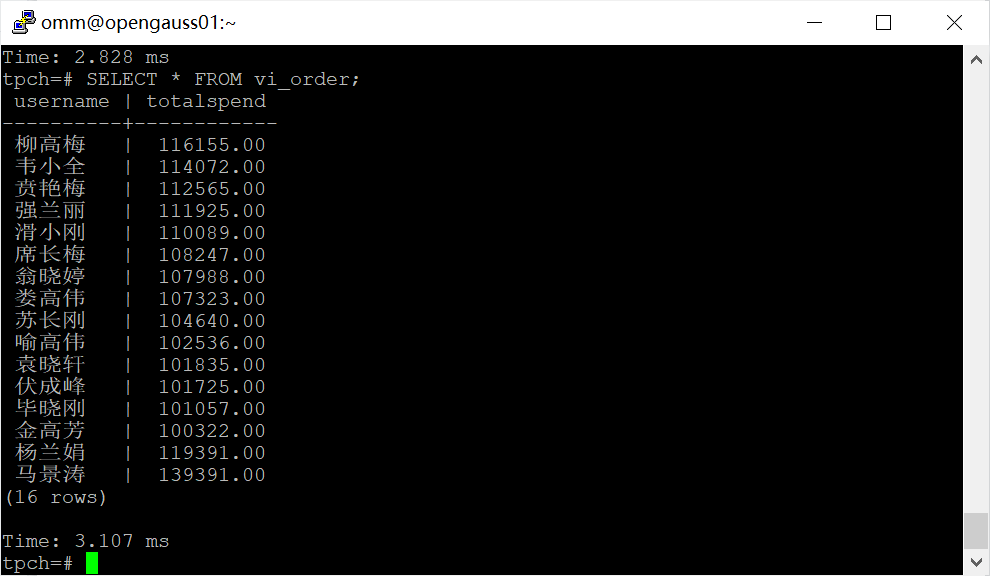
SELECT \* FROM vi\_order;



5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;





实践思考题1：行存表与列存表在执行相同的SQL语句时，为何执行的时间不同？在执行哪些类型SQL时，行存表效率更高？在执行哪些类型SQL时，列存表效率更高？

在执行相同的SQL语句时，它们的执行时间可能会有差异。这是由于它们在数据存储和读取方面的设计差异导致的。  
  
行存表（Row-based table）将每一行数据作为一个存储单元，并按照列的顺序依次存储。当执行需要读取整行数据的SQL语句时，行存表效率较高，因为它可以快速地获取一条完整的记录。  
  
列存表（Column-based table）将每一列数据作为一个存储单元，并按照列的顺序进行存储。当执行需要读取某些列数据的SQL语句时，列存表效率较高，因为它可以仅读取需要的列，而无需读取其他列的数据，从而减少了数据的读取量。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

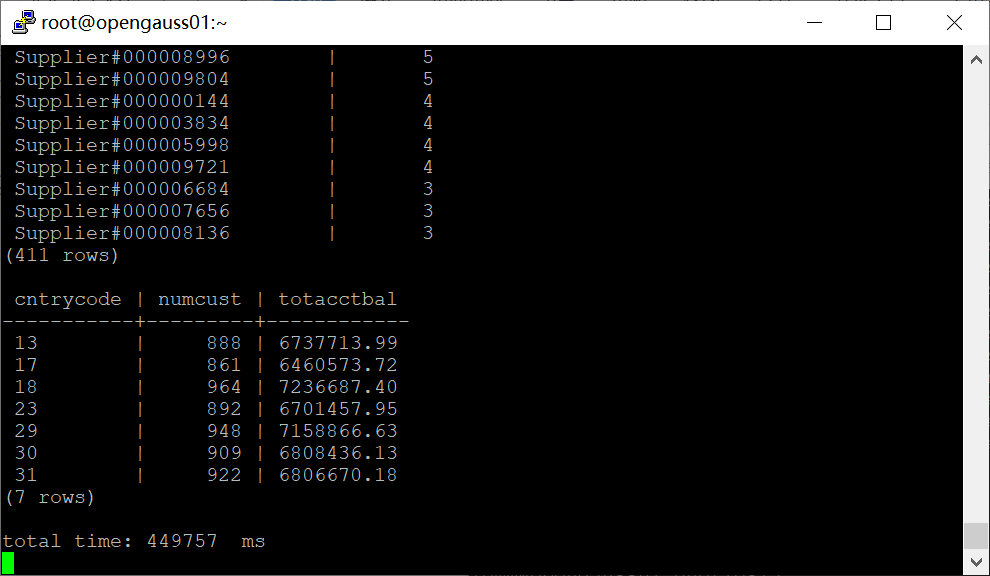
全量物化视图和增量物化视图的主要差别在于：  
  
数据更新方式：全量物化视图需要重新计算和加载所有数据，而增量物化视图只更新变化的部分数据。  
刷新效率：增量物化视图由于仅处理变化的数据，因此刷新效率通常更高。  
实现方式：增量物化视图可能需要额外的日志或触发器等机制来跟踪数据变化，而全量物化视图只需要在特定的时间点或条件下重新计算和加载数据。

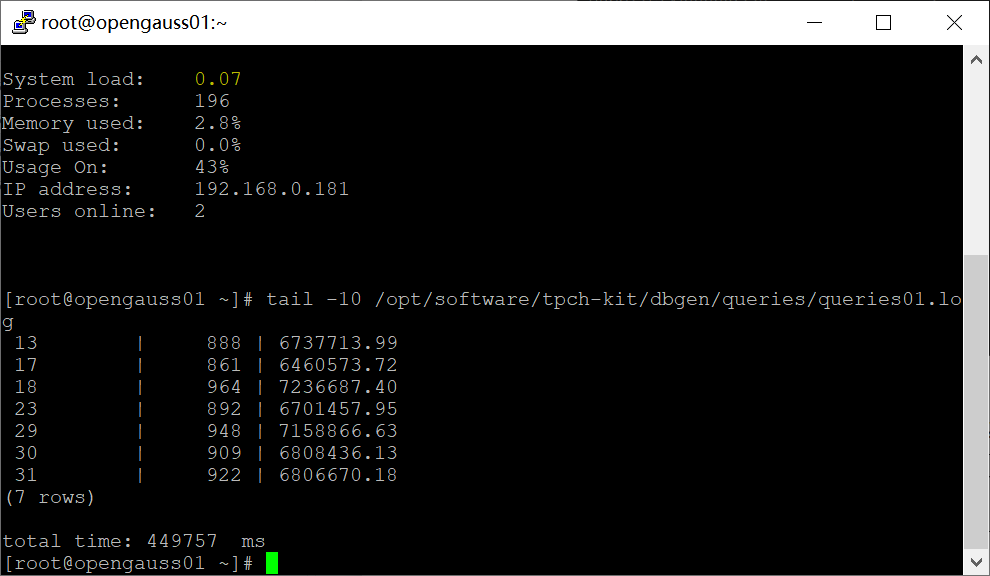
# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

1. 执行TPCH脚本，获得测试时间，将执行结果截图：

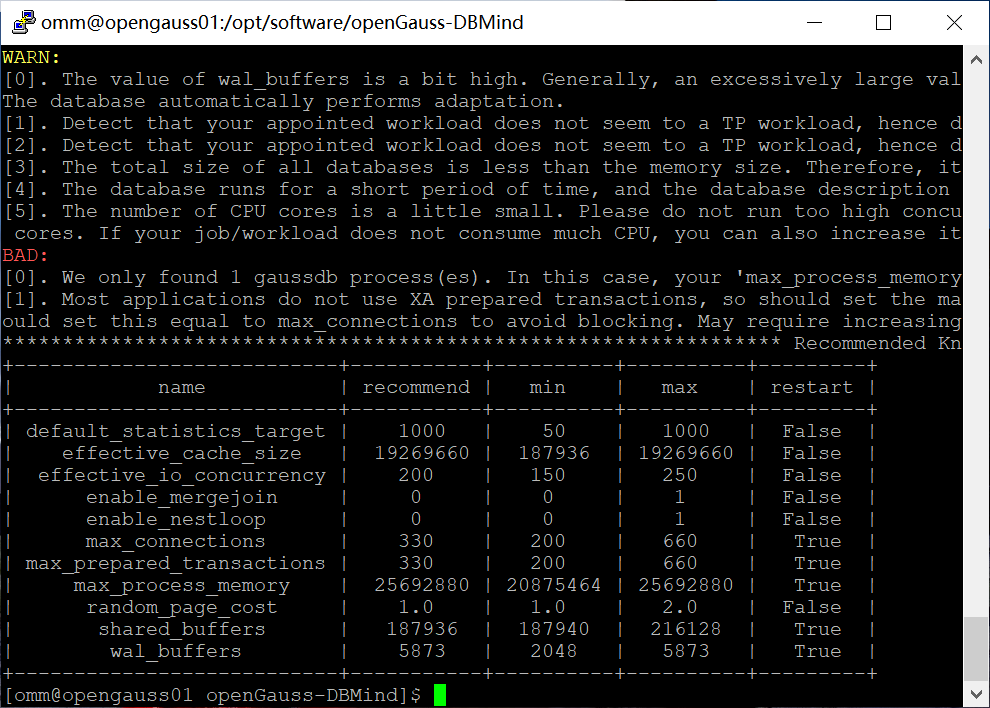
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





2. 使用root用户，执行X-Tuner进行参数建议优化，将执行结果截图

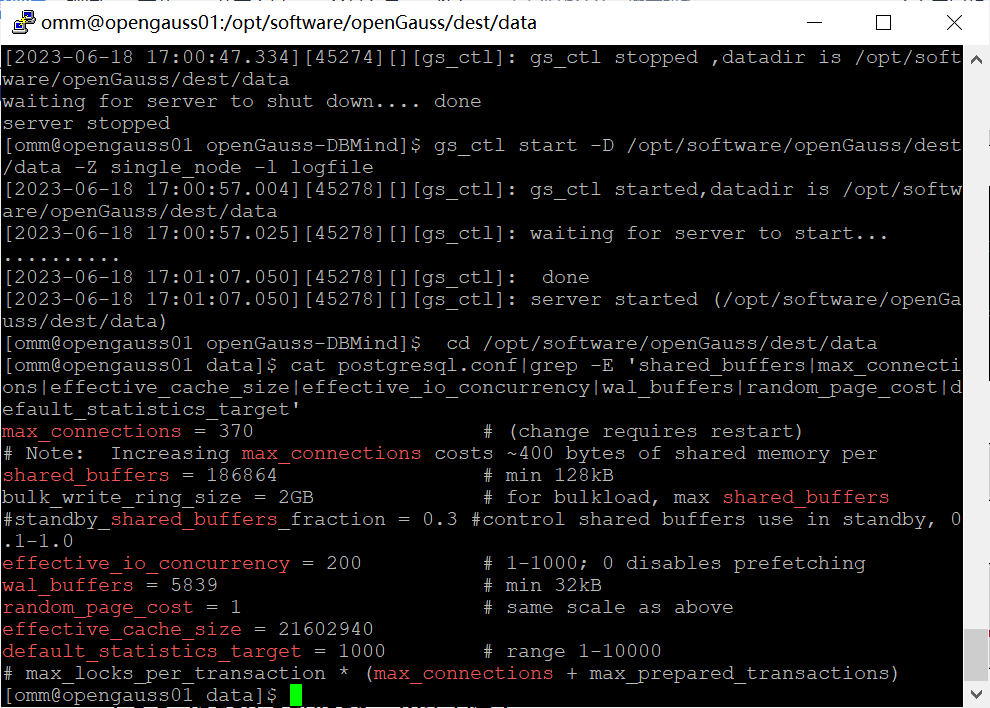
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



3.重启完成后，获取参数值：

cd /opt/software/openGauss/data

cat postgresql.conf|grep -E 'shared\_buffers|max\_connections|effective\_cache\_size|effective\_io\_concurrency|wal\_buffers|random\_page\_cost|default\_statistics\_target'



任务二：使用Index-advisor对select 查询语句进行优化，并通过对比执行计划，得到优化前后的不同。

1. 使用explain，对查询2020年3月订单表收入并进行排序的SQL加以分析，将结果截图。

EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

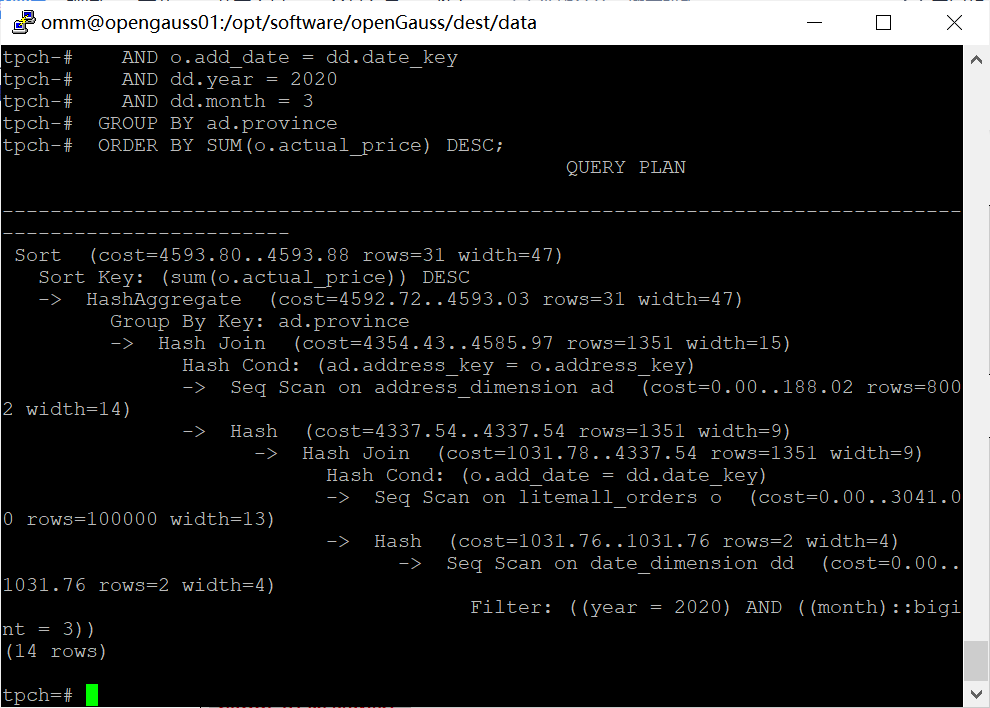
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



2. 使用索引推荐功能，对查询语句进行推荐，将执行结果截图。

select \* from gs\_index\_advise('

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

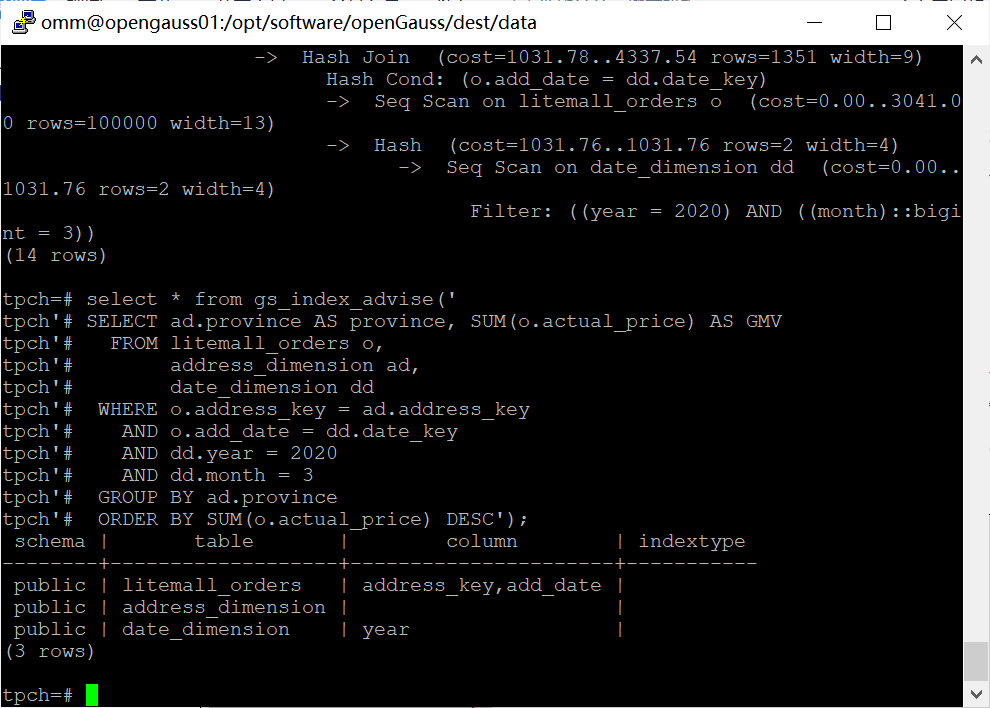
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

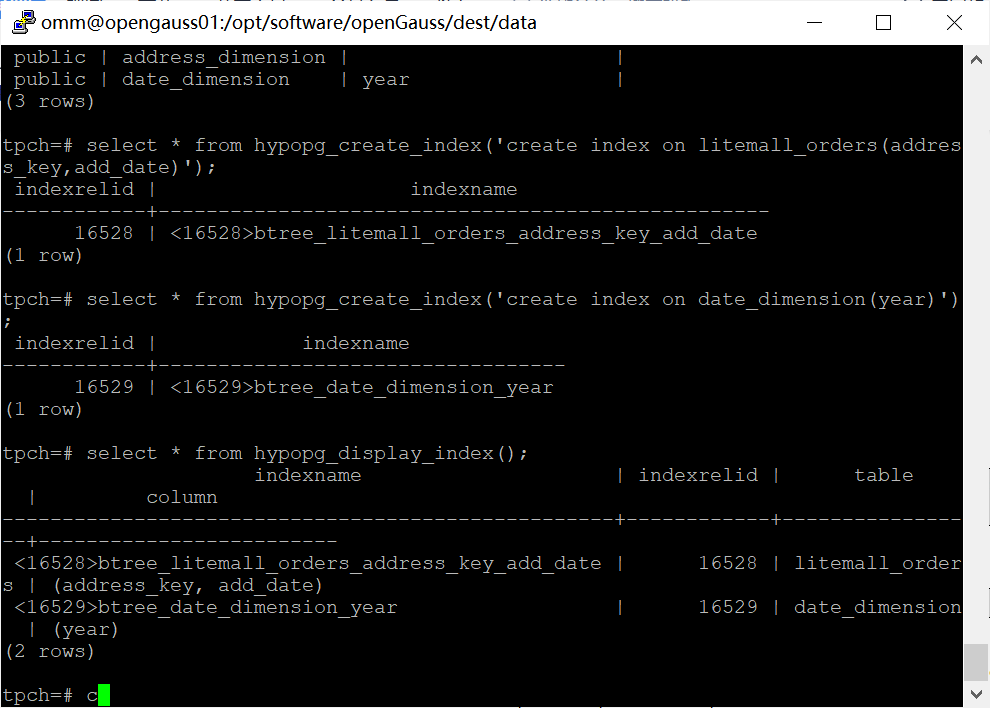
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC');



3. 查看创建的虚拟索引列，将执行结果截图。

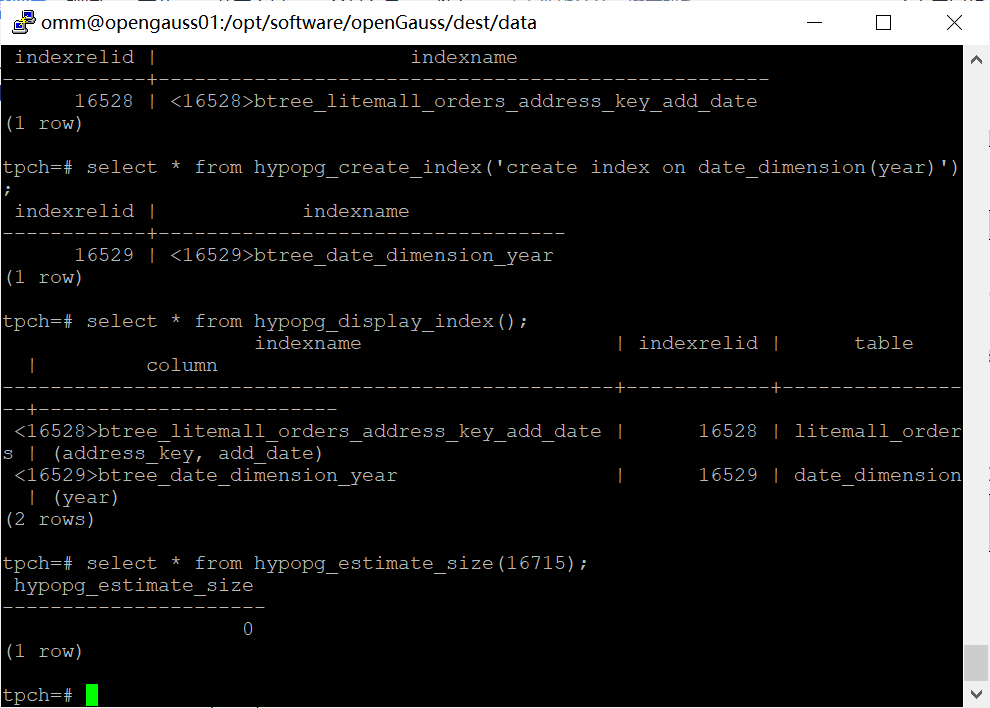
select \* from hypopg\_display\_index();

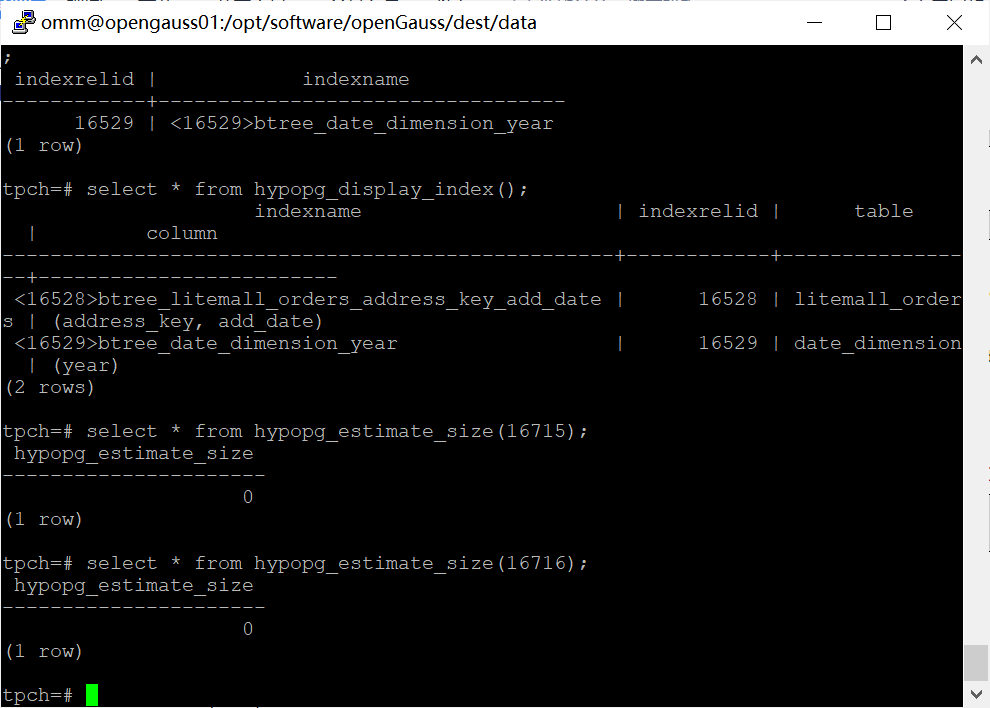


4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

select \* from hypopg\_estimate\_size(16715);

select \* from hypopg\_estimate\_size(16716);





5.再次使用explain，对该SQL加以分析，将执行结果截图。

EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

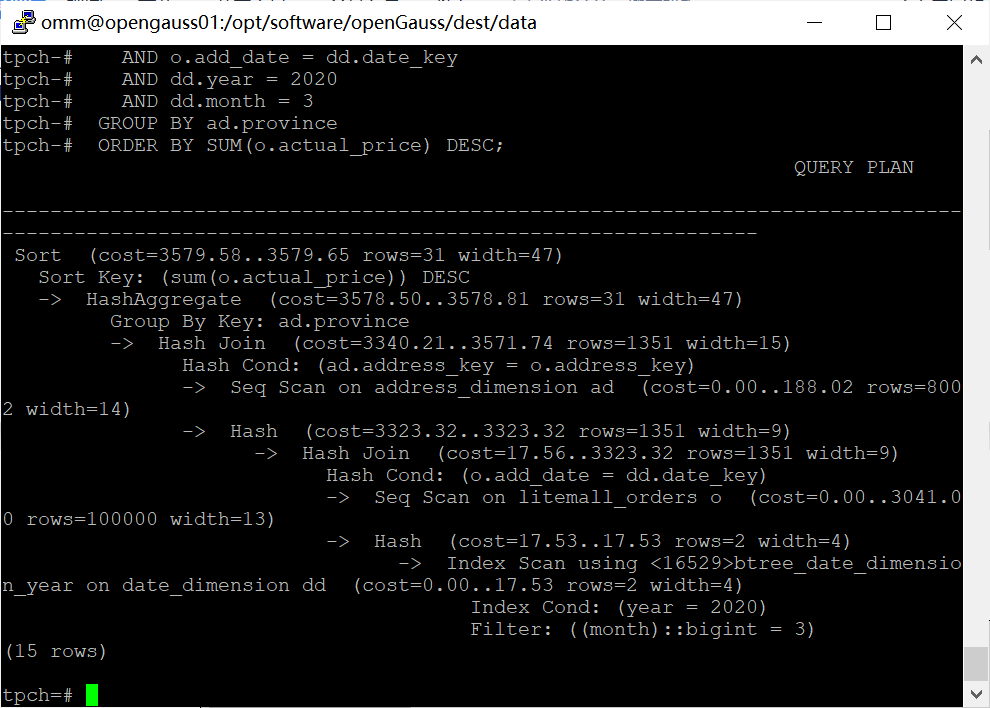
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

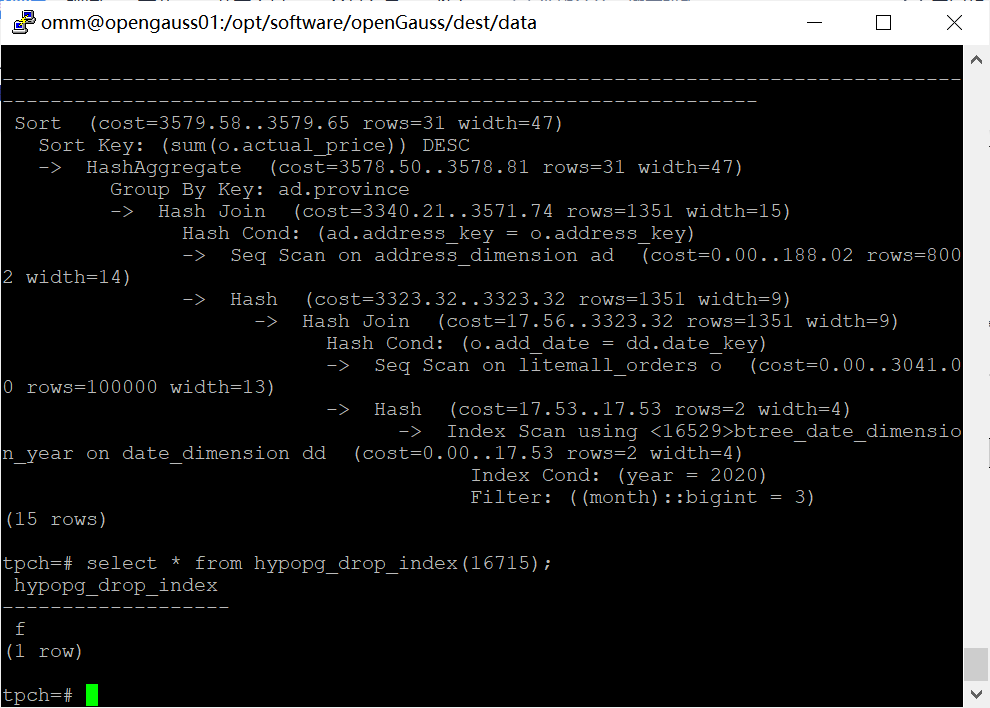
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



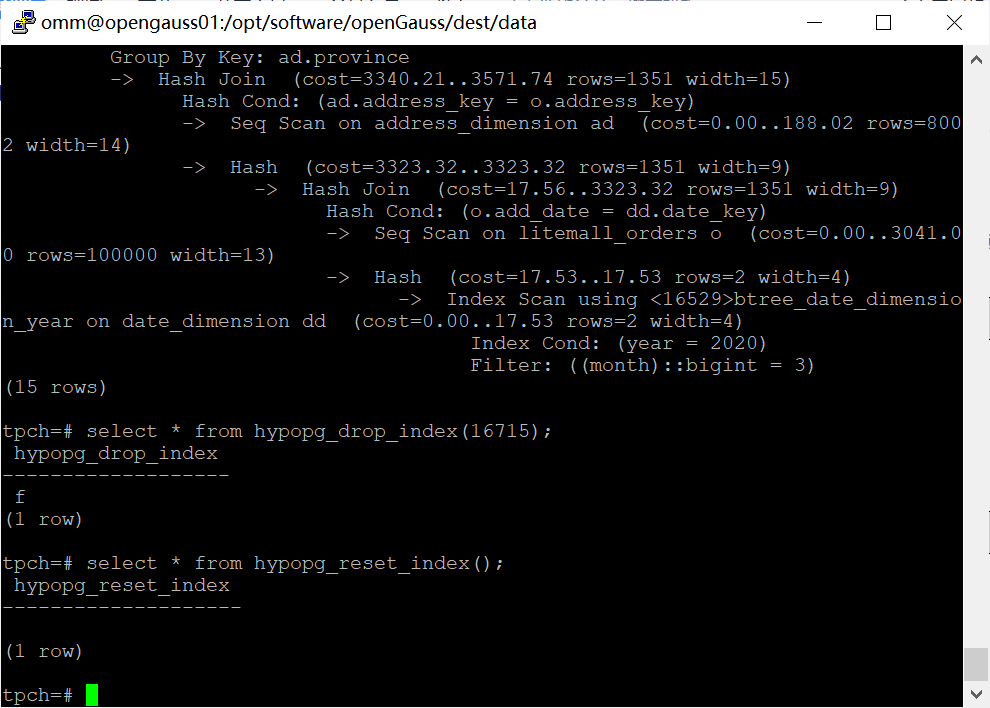
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(16715);



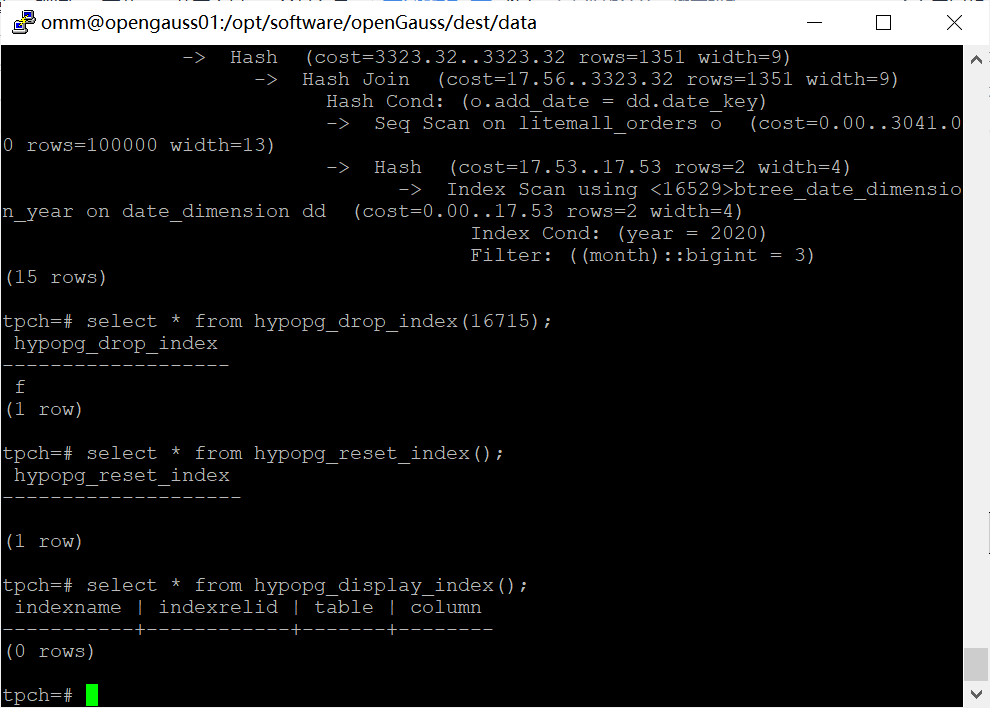
7. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_reset\_index();



8. 查看索引虚拟列，将执行结果截图。

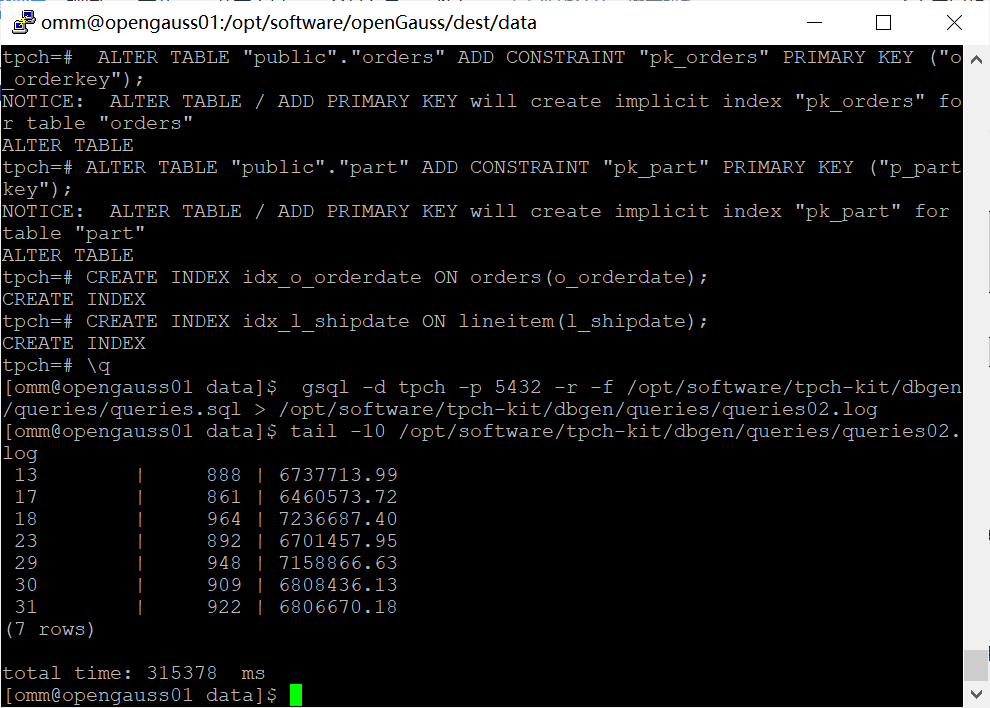
select \* from hypopg\_display\_index();



任务三：通过创建索引，对queries.sql中的SQL语句进行优化，并对比优化前后queries.sql执行的时间。

1. 重新执行queries.sql查询，将执行结果截图：

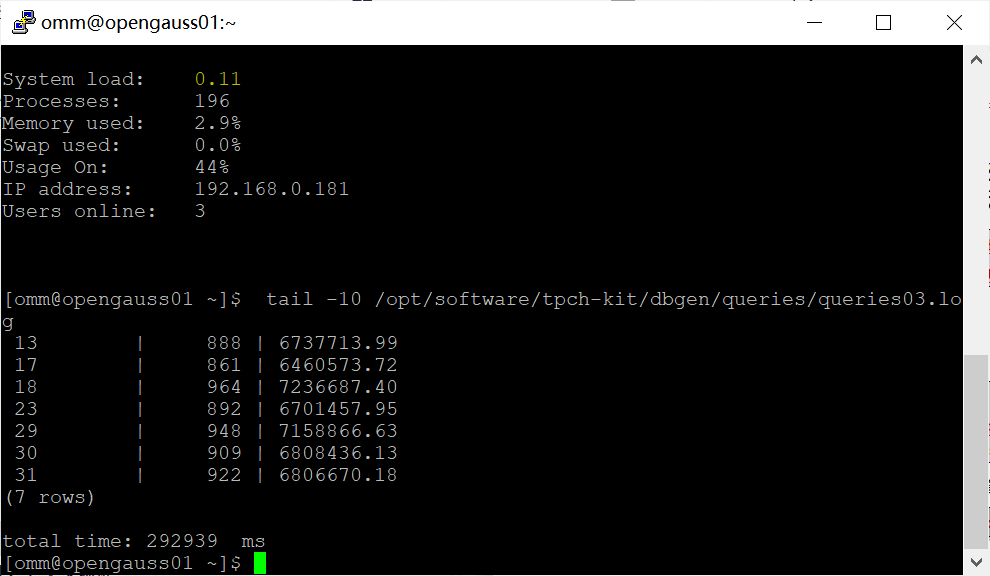
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

1. 重新执行queries.sql查询，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries03.log



实践思考题1：根据X-Tuner给出的参数优化，在哪些参数上进行了优化，为何要对这些参数进行优化？

对shared\_buffers、max\_connections、effective\_cache\_size、effective\_io\_concurrency、wal\_buffers、random\_page\_cost、default\_statistics\_target参数进行了优化。

1. shared\_buffers（共享缓冲区）：  
   - 优化目标：共享缓冲区用于存储从磁盘读取的数据块，提高数据库的性能。通过优化该参数，可以使数据库更有效地使用内存，减少磁盘IO，提高查询性能。  
   - 优化原因：增加共享缓冲区大小可以提高数据缓存的命中率，减少从磁盘读取数据的次数，加快查询和数据访问的速度。  
2. max\_connections（最大连接数）：  
   - 优化目标：最大连接数指定了数据库系统同时支持的最大客户端连接数。通过优化该参数，可以确保系统具有足够的连接资源来处理并发请求，避免资源瓶颈和性能下降。  
   - 优化原因：如果并发连接数超过了数据库系统的处理能力，可能会导致连接排队、延迟和性能下降。适当增加最大连接数可以提高系统的并发处理能力。  
3. effective\_cache\_size（有效缓存大小）：  
   - 优化目标：有效缓存大小表示数据库可以使用的系统缓存大小。通过优化该参数，可以告诉查询优化器有关系统缓存大小的信息，以便它可以更好地估计查询的成本和选择合适的查询执行计划。  
   - 优化原因：查询优化器使用有效缓存大小来估计查询的成本和选择索引或顺序扫描等查询执行计划。正确配置有效缓存大小可以提高查询计划的准确性，从而提高查询性能。  
4. effective\_io\_concurrency（有效IO并发度）：  
   - 优化目标：有效IO并发度表示数据库在执行IO操作时可以并行进行的数量。通过优化该参数，可以提高并发IO操作的效率，减少IO等待时间，提高查询和数据访问的速度。  
   - 优化原因：数据库通常需要执行许多IO操作，如从磁盘读取数据块。通过增加有效IO并发度，可以让数据库并行执行多个IO操作，从而提高整体IO吞吐量和响应速度。  
5. wal\_buffers（WAL缓冲区）：  
   - 优化目标：WAL缓冲区用于存储事务日志（Write-Ahead Log），用于持久化数据修改操作。通过优化该参数，可以提高WAL写入的性能，减少事务提交的延迟。  
   - 优化原因：较大的WAL缓冲区可以减少频繁的磁盘写入操作，提高写入性能和持久化的效率  
。较小的WAL缓冲区可能会导致频繁的磁盘写入操作，降低事务提交的速度。  
6. random\_page\_cost（随机页面成本）：  
   - 优化目标：随机页面成本表示随机IO操作相对于顺序IO操作的成本。通过优化该参数，可以影响查询优化器在选择查询执行计划时对随机IO成本的考虑程度。  
   - 优化原因：随机IO通常比顺序IO更加昂贵，因为它需要在磁盘上进行磁头的寻道操作。通过适当调整随机页面成本，可以让查询优化器更准确地估计不同查询执行计划的成本，从而选择更高效的查询计划。  
7. default\_statistics\_target（默认统计目标）：  
   - 优化目标：默认统计目标用于指定自动生成表和索引的统计信息的精度级别。通过优化该参数，可以提高查询优化器对表和索引数据的统计分析准确性，从而生成更好的查询执行计划。  
   - 优化原因：查询优化器使用统计信息来估计查询成本、选择索引和执行计划。通过增加默认统计目标，可以增加生成统计信息的详细程度，提高查询优化器的决策准确性，从而提高查询性能。

实践思考题2：索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

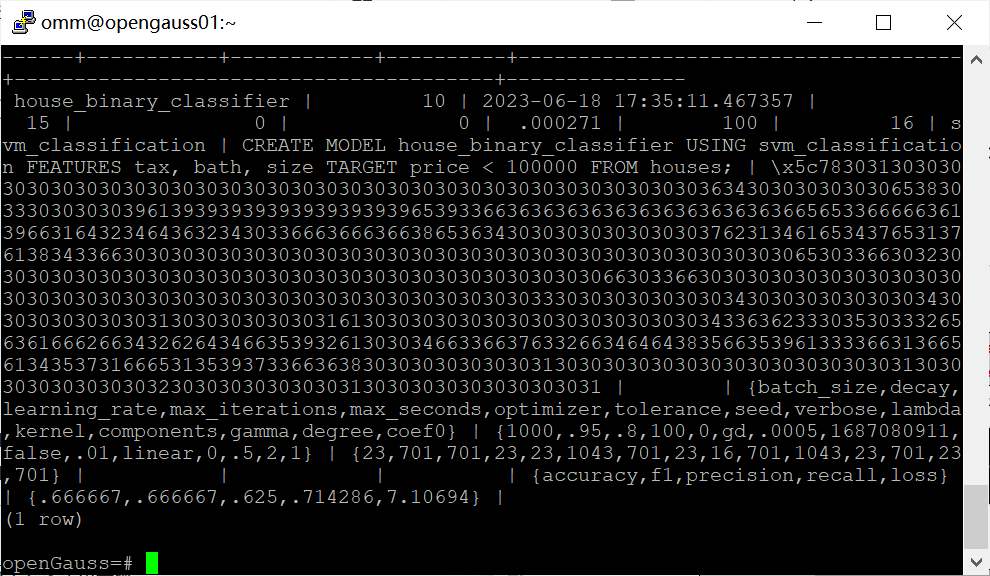
快速数据检索：索引可以提高数据的检索速度。通过在关键列上创建索引，数据库可以直接定位到满足查询条件的数据，而不需要完全扫描整个表。  
  
减少磁盘IO：索引可以减少磁盘IO操作的次数。通过使用索引，数据库可以更快地读取和返回所需数据，减少物理读取的次数和数据传输的开销。  
  
提升查询性能：索引可以加速查询语句的执行速度。通过优化查询的执行计划，数据库可以更高效地执行查询操作，从而提升整体的查询性能。

除了使用索引和参数来优化数据库性能，还有以下方面可以进行数据库优化：  
  
数据库设计：优化数据库的物理和逻辑设计，包括表的结构设计、数据类型选择、范式规范等。合理设计数据库模式可以提高数据的存储效率和查询性能。  
  
查询优化：通过优化查询语句的编写和执行计划的选择，可以提高查询性能。使用适当的JOIN语句、子查询、索引提示等技术可以优化查询的执行效率。  
  
数据库索引优化：除了创建适当的索引，还需要定期维护和优化索引。这包括重新组织索引、删除不使用的索引、调整索引的列顺序等。  
  
缓存和缓冲区管理：使用适当的缓存机制，如查询结果缓存、数据库缓冲区等，可以减少磁盘IO操作，提高数据库的访问速度。  
  
合理配置硬件和网络：对于大型数据库系统，合理配置硬件资源和网络环境可以提升数据库的性能和可靠性。这包括内存大小、磁盘配置、网络带宽等的优化。  
  
定期维护和监控：定期进行数据库的备份、恢复和优化操作，监控数据库的性能和运行状态，及时发现和处理潜在问题。

# 关卡四、openGauss的DB4AI特性应用

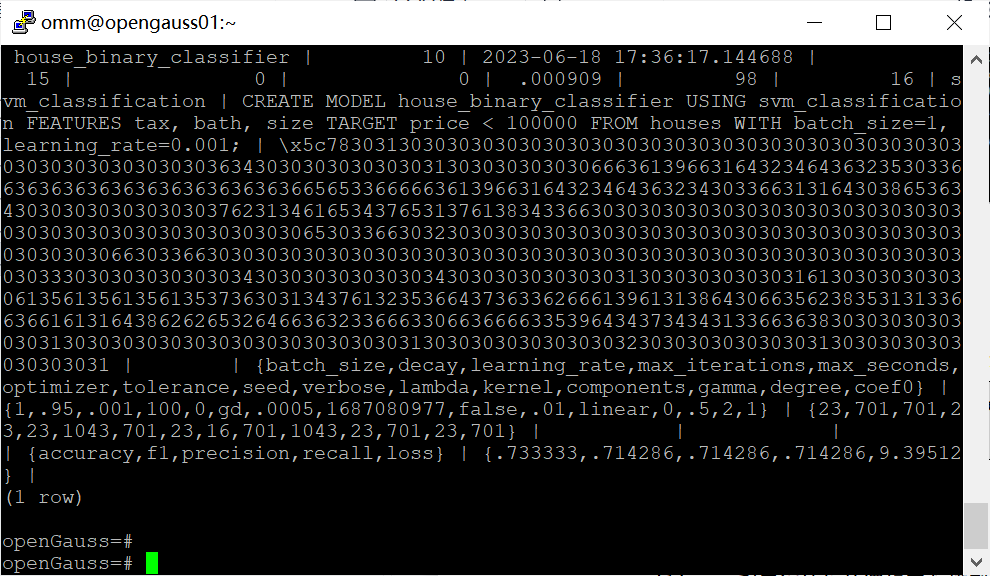
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



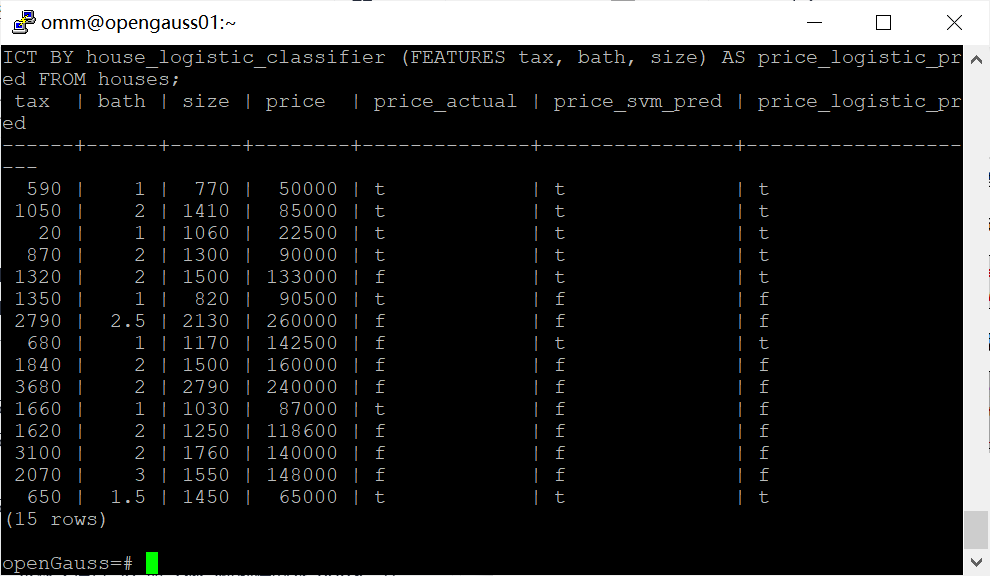
任务二：观察新模型的信息，将执行结果截图。

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



任务三：利用训练好的逻辑回归模型预测数据，并与SVM算法进行比较，将执行结果截图。

postgres=# SELECT tax, bath, size, price, price < 100000 AS price\_actual, PREDICT BY house\_binary\_classifier (FEATURES tax, bath, size) AS price\_svm\_pred, PREDICT BY house\_logistic\_classifier (FEATURES tax, bath, size) AS price\_logistic\_pred FROM houses;



实践思考题1：分类模型与回归模型有何不同？

分类模型和回归模型是机器学习中两种常见的预测模型，它们在目标变量类型和输出结果上有所不同。  
  
目标变量类型：  
  
分类模型的目标变量是离散型的，代表了不同的类别或标签。例如，预测一个电子邮件是垃圾邮件还是非垃圾邮件、预测病人是否患有某种疾病等。  
回归模型的目标变量是连续型的，代表了一个连续或数值型的结果。例如，预测房屋的价格、预测销售额等。  
输出结果：  
  
分类模型输出的是分类标签或预测的类别。例如，分类模型可以输出二元标签（如垃圾邮件/非垃圾邮件）或多元标签（如猫/狗/鸟等）。  
回归模型输出的是一个连续值，可以是一个实数或者一个范围。例如，回归模型可以预测一个数值，如预测房屋价格为120,000美元，也可以预测一个范围，如房屋价格在100,000到150,000美元之间。  
模型选择和评估：  
  
分类模型通常使用分类算法来进行训练和预测，比如逻辑回归、决策树、支持向量机（SVM）等。评估分类模型的性能常用的指标包括准确率、召回率、F1得分等。  
回归模型通常使用回归算法来进行训练和预测，比如线性回归、决策树回归、随机森林回归等。评估回归模型的性能常用的指标包括均方误差（MSE）、平均绝对误差（MAE）等。

实践思考题2：什么是SVM算法？

支持向量机（Support Vector Machine，简称SVM）是一种经典的监督学习算法，用于分类和回归问题。它的目标是在特征空间中找到一个最优的超平面，将不同的实例点分隔开来。  
  
SVM的核心思想是通过将数据映射到高维特征空间，使得不同类别的数据点在该空间中能够更好地分离。在特征空间中，SVM寻找一个最优超平面，该超平面能够最大化不同类别之间的间隔（Margin）。这个最优超平面被称为分隔超平面（Separating Hyperplane），具有最大间隔的特点，因此也被视为最优分类边界。

实践思考题3：分类问题有哪些评价指标，请分别说明他们的含义？

在分类问题中，常用的评价指标包括准确率（Accuracy）、精确率（Precision）、召回率（Recall）、F1值（F1-Score）和ROC曲线与AUC。  
  
准确率（Accuracy）：  
准确率是最简单直观的评价指标，它表示分类正确的样本数占总样本数的比例。准确率较高时，表示模型的总体预测能力较好。然而，当不同类别的样本数量不平衡时，准确率可能会产生误导性的结果。  
  
精确率（Precision）：  
精确率是指在所有被预测为正例的样本中，真正正例的比例。它衡量了模型对于正例的判断能力。精确率较高时，表示模型将负例误判为正例的概率较低。  
  
召回率（Recall）：  
召回率是指在所有实际为正例的样本中，被正确预测为正例的比例。它衡量了模型对于正例的识别能力。召回率较高时，表示模型能够较好地找到正例。  
  
F1值（F1-Score）：  
F1值是精确率和召回率的调和平均值，综合了两者的指标。它能够综合考虑模型的精确度和召回率，适用于样本不平衡情况。  
  
ROC曲线与AUC：  
ROC曲线是以假正例率（False Positive Rate）为横轴，真正例率（True Positive Rate）为纵轴绘制的曲线。ROC曲线能够展现分类器在不同阈值下的性能表现。AUC（Area Under Curve）是ROC曲线下的面积，用来衡量分类器的性能。AUC值越大，表示分类器的性能越好。

实践思考题4：回归问题有哪些评价指标，请分别说明他们的含义？

在回归问题中，常用的评价指标包括均方误差（Mean Squared Error，MSE）、均方根误差（Root Mean Squared Error，RMSE）、平均绝对误差（Mean Absolute Error，MAE）、决定系数（Coefficient of Determination，R-squared）等。  
  
均方误差（Mean Squared Error，MSE）：  
均方误差是回归问题中最常用的评价指标之一，它表示预测值与真实值之间差异的平方的平均值。MSE值越小表示模型的预测越准确。  
  
均方根误差（Root Mean Squared Error，RMSE）：  
均方根误差是均方误差的平方根，它与原始数据的单位相同。RMSE在评价模型预测误差时更容易理解，值越小表示模型的预测越准确。  
  
平均绝对误差（Mean Absolute Error，MAE）：  
平均绝对误差是预测值与真实值之间差异的绝对值的平均值。MAE衡量了模型预测的平均偏差程度，值越小表示模型的预测越准确。  
  
决定系数（Coefficient of Determination，R-squared）：  
决定系数是用来衡量回归模型对观测数据拟合程度的指标，它表示由模型解释的目标变量方差的比例。R-squared的取值范围为0到1，越接近1表示模型对目标变量的解释能力越强。